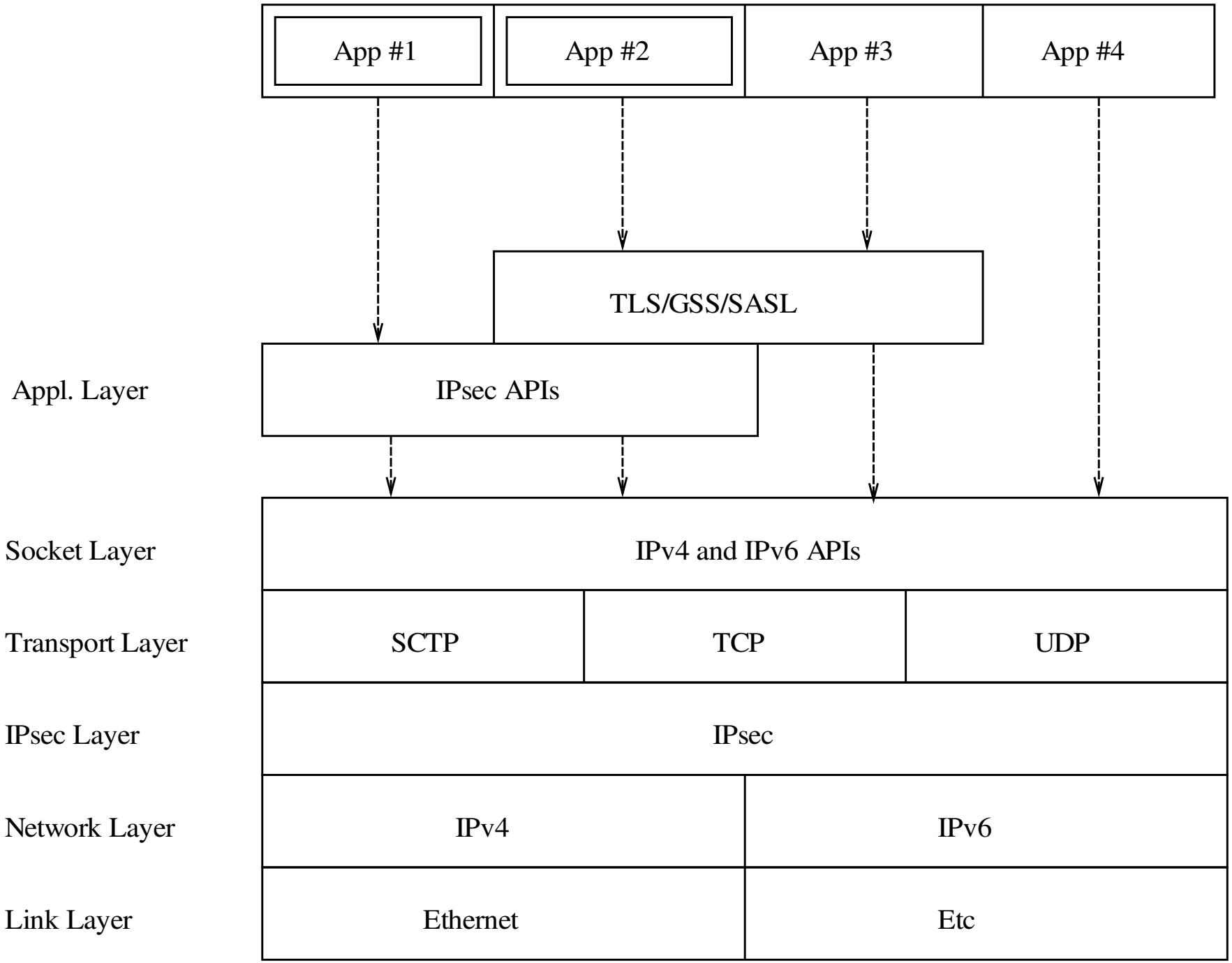
C-Bindings for BTNS APIs

Miika Komu <miika@iki.fi>
Sasu Tarkoma <sasu.tarkoma@hiit.fi>
Nicolas Williams <nicolas.williams@sun.com>
Michael Richardson <
mcr@sandelman.ottawa.on.ca>

What Problem Are We Solving?

- How does an network application know that a connection is secured by IPsec?
- How can the application tell explicitly that the use of BTNS extensions is ok?
- How to this in a portable way in the sockets API in C-language?


App #1
App #2
App #3
App #4
TLS/GSS/SASL
Appl. Layer
IPsec APIs
Socket Layer
IPv4 and IPv6 APIs
Transport Layer
SCTP
TCP
UDP
IPsec Layer
IPsec
Network Layer
IPv4
IPv6
Link Layer
Ethernet
Etc

Relationship to GSS/SASL

- GSS/SASL APIs deal with upper layer security
  - GSS/SASL APIs are not based on socket descriptors
- IPsec APIs deal with lower layer security
  - IPsec can be used without any changes in the application
  - IPsec APIs are based on socket descriptors
- IPsec APIs can be used in an application simultaneously with GSS or SASL

Policies

```
typedef ipsec_policy_t struct ipsec_policy;

ipsec_policy_t *ipsec_create_policy(uint32_t type);
int ipsec_free_policy(ipsec_policy_t *policy);

int ipsec_get_policy_attr(const ipsec_policy_t *policy,
                          uint32_t attr_type,
                          uint32_t *attr_len,
                          void **attr_val);
int ipsec_set_policy_attr(ipsec_policy_t *policy,
                          uint32_t attr_type,
                          uint32_t attr_len,
                          const void *attr_val);

int ipsec_set_socket_policy(int fd, const ipsec_policy_t *policy);
int ipsec_get_socket_policy(int fd, ipsec_policy_t **policy);

int ipsec_get_policy_attr(const ipsec_policy_t *policy,
                          uint32_t attr_type,
                          uint32_t *attr_len,
                          void **attr_val);
int ipsec_set_policy_attr(ipsec_policy_t *policy,
                          uint32_t attr_type,
                          uint32_t attr_len,
                          const void *attr_val);
```

Channel Bindings

```
typedef ipsec_channel_info_t struct ipsec_channel_info;

ipsec_channel_info_t *ipsec_create_channel_info();
int ipsec_free_channel_info(*ipsec_channel_info_t *ci);

int ipsec_set_channel_attr(ipsec_channel_info_t *ci,
                           uint32_t attr_type,
                           uint32_t attr_len,
                           const void *attr_val);
int ipsec_get_channel_attr(const ipsec_channel_info_t *ci,
                           uint32_t attr_type,
                           uint32_t *attr_len,
                           void **attr_val);
int ipsec_set_socket_channel_info(int fd,
                                  constipsec_channel_info_t *ci);
int ipsec_get_socket_channel_info(int fd,
                                  ipsec_channel_info_t **ci);
int ipsec_set_msg_channel_info(const struct *msg_hdr,
                               const ipsec_channel_info_t *ci);
int ipsec_get_msg_channel_info(const struct *msg_hdr,
                               ipsec_channel_info_t **ci);
int ipsec_cmp_channel_info(const ipsec_channel_info_t *ci1,
                           const ipsec_channel_info_t *ci2);
int ipsec_dup_channel_info(const ipsec_channel_info_t *ci,
                           ipsec_channel_info_t **ci_dup);
```

ChangeLog from 00 to 01

- 00 included only ideas, but 01 contains concrete API definitions
  - Hello world applications in the appendix
- Based on comments from Nicolas Williams, Michael Richardson, Love Åstrand and Julien Laganier

Todo list 1/3

- SASL/GSS code examples
- Storing of channel bindings to disk
  - Binary descriptions (similar to GSS_export)
- Querying of local and peer identities
- Error values
- Channel binding is not a settable thing
- Associate channel bindings with protection tokens, not with sockets
- Remove “prevent IKE authentication attribute”, replace with BTNS OK

Todo list 2/3

- rename: ipsec_policy -> protection token
  - contains information on e.g. IPsec algos
- add: identity token
  - local or peer public identities
- add: credential token
  - private or secret keys, PIN number prompt, etc
- constants should be strings in the API
  - channel bindings should be octet strings
- typedefs should be pointers
- common attribute accessors

Todo list 3/3

- attribute1 < attribute2
- Define attributes for protection tokes
  - KE protocol: EAP, PKIX, BTNS, HIP
  - algo profile names, lifetimes
- Define attributes for identity tokens
  - ID type, name, certificates
- Conversion functions: human readable/loggable protection and identity tokens
- Some editorial corrections